# How significant is the initial population when using a genetic algorithm to solve the quadratic assignment problem?

Tianzhu Liu[1]

[1]*Department of Mathematics, Bucknell University, Lewisburg, PA 17837*

*Project Advisor: Lucas Waddell, Assistant Professor of Mathematics, Bucknell University*

August 18, 2021

**Abstract**

The quadratic assignment problem (QAP) is perhaps the most widely studied discrete nonlinear optimization problem, due both to its many practical applications as well as the difficulty associated with solving it. One popular approach to efficiently find good solutions to large instances of the QAP is to use genetic algorithms (GAs), which are metaheuristics based on Charles Darwin's theory of evolution and his idea of "survival of the fittest." One of the most important features of a genetic algorithm is the initial population. Some practitioners prefer to use completely random initial populations, while others prefer to "seed" the initial population with good heuristic solutions. While the choice of initial population method can typically have a significant impact on the performance of a GA, previous work has not clearly suggest whether GA has significant effect when solving the QAP. In this study, we provide evidence that the choice of method for creating the initial population does not affect (in a statistically significant way) the solution obtained by a GA when solving the QAP.

## 1 Introduction

In this paper, we investigate how the method used to create the initial population affects the performance of a genetic algorithm (GA) when used to solve the quadratic assignment problem (QAP). Section 1.1 provides background on the QAP, while Section 1.2 introduces GAs and presents our research question. Section 2 gives an overview of our methodology, with Section 2.1 detailing the structure of our GA. Our computational results are discussed in Section 3, and Section 4 contains concluding remarks. The Python code for our GA is provided in the Supplemental Material.

## 1.1 The Quadratic Assignment Problem (QAP)

The QAP is an NP-hard combinatorial optimization problem that can be formulated as

$$\text{QAP: } \min_{\phi \in S_n} \left( \sum_{i=1}^{n} c_{\phi(i)i} + \sum_{i=1}^{n} \sum_{j=1}^{n} f_{\phi(i)\phi(j)} d_{ij} \right), \tag{1}$$

where $S_n$ is the set of all permutations of the numbers 1 through $n$. The objective is to find the solution (out of all $n!$ possible permutations) that minimizes the given cost function.

The QAP was first introduced in the setting of facility location problems in the 1950s by two economists, Tjalling Koopmans and Martin Beckmann [11]. To understand this context, imagine a company that wishes to construct $n$ different facilities, one on each of $n$ different location sites, in the least expensive way. Notationally, each possible assignment is represented by a permutation $\phi \in S_n$. For example, if there are $n = 4$ facilities and location sites, then the permutation $\phi = (3,1,4,2)$ represents the solution where facility 3 is assigned to location site 1, facility 1 to site 2, facility 4 to site 3, and facility 2 to site 4. The objective is to minimize the total cost of construction and material flow. The construction cost of building facility $\phi(i)$ on location site $i$ is given by $c_{\phi(i)i}$. The cost of material flow between location sites $i$ and $j$ is given by $f_{\phi(i)\phi(j)} d_{ij}$, where $d_{ij}$ is the distance between sites $i$ and $j$, and $f_{\phi(i)\phi(j)}$ is the amount of material flowing between facilities $\phi(i)$ and $\phi(j)$. The addition of the $f_{\phi(i)\phi(j)} d_{ij}$ term to the cost function ensures that facilities that interact with each other frequently will tend to be built closer to each other than facilities with relatively infrequent interaction.

The name *quadratic assignment problem* arises from the fact that an alternate binary programming formulation of the QAP (see [2], for example) involves minimizing a quadratic function of binary decision variables over the assignment polytope. Note that in the formulation (1), non-symmetric flows (i.e., $f_{ij} \neq f_{ji}$) can be used to account for directed material flow data, while non-symmetric distances can account for situations involving, say, one-way streets or upstream/downstream shipping effects. Additionally, a more general form of the QAP exists where the cost terms in the double-sum do not possess a flow-times-distance structure. In that case, the term simply represents the cost associated with simultaneously building facility $\phi(i)$ on site $i$ *and* building facility

$\phi(j)$ on site $j$. In recognition of [11], whenever an instance of the QAP *does* possess the flow-times-distance structure of (1), it is said to be in "Koopmans-Beckmann form." Surveys on the QAP are available in [2], [5], and [13].

Over the years, the QAP has been applied to areas as diverse as college campus planning [7], hospital layout [8], typewriter keyboard design [4], and assigning airplanes to gates at airport terminals [9], just to name a few. Unfortunately, despite its many applications, the QAP has proven to be incredibly difficult to solve using exact solution m ethods. This has caused operations research practitioners to use metaheuristic algorithms (see, for example, [1, 14, 15]) on larger instances of the QAP. Metaheuristic algorithms, such as GAs, simulated annealing, and tabu search, seek to efficiently explore the search space of an optimization problem in order to provide a sufficiently good, but not provably optimal, solution in a reasonable amount of time. In this study, we focus on the use of GAs to solve the QAP.

## 1.2 Genetic Algorithms (GAs)

GAs are metaheuristics that are based on Charles Darwin's theory of evolution, particularly his ideas of natural selection and survival of the fittest. Applied to an optimization problem like the QAP, a GA begins with an initial *population* of potential solutions to the problem (specifically, some collection of permutations from $S_n$). These solutions undergo small changes (called *mutations*) and combination operations (called reproduction or *crossover*) to form new solutions, the best of which (those with the most *fitness*, namely the permutations with the lowest cost or the highest efficiency) are chosen to form the next *generation* of solutions. This evolutionary process is repeated many times, with the basic idea being that the quality of the population will improve from generation to generation, and will eventually consist of optimal or near-optimal solutions.

Although the basic structure of a GA is always the same, there are many choices to make and variations to consider when it comes to implementation. In Section 2.1 we discuss in more detail the choices that we made for our GA. One of the many decisions that needs to be made is how the initial population will be constructed. Some common choices are a completely random initial population, an initial population that is "seeded" with solutions that are known to have good fitness

(in the case of the QAP, this means a small cost), or some combination of these two strategies. It is widely known for some problems, the choice of initial population can greatly affect a GA's ability to converge to near-optimal solutions (see, for instance, [6] and its references).

Relative to the QAP, the paper [1] has the most thorough treatment of initial population methods. They note that "the performance of a GA is often sensitive to the quality of its initial population. The 'goodness' of the initial population depends both on the average fitness (that is, the objective function value) of individuals in the population and the diversity in the population. Losing on either count tends to produce a poor GA." Later, in the computational results section of their paper, they briefly mention that they "observed from this limited computational testing that the performance of the GA is relatively insensitive to the method used for initial population generation." They did, however, report small performance differences when using different initial population schemes. Since there is so much randomness inherent in a GA, this begs the question of whether or not these performance differences are statistically significant. Furthermore, the authors of [1] only tested three different *good* initial population methods. We wanted to extend this study to also include *bad* initial populations to definitively answer our proposed research question: **Do different initial population methods produce statistically significantly different results when using a GA to solve the QAP?**

## 2 Methodology

In order to answer our research question, we first needed to design a GA to use in our experiments. As mentioned before, there are many choices to make when designing a GA. Our specific choices are detailed in Section 2.1 and illustrated in Figure 1. Section 2.1 also describes the five different initial population methods that we used in our tests.

We performed our experiments using seven different QAP test instances from an online QAP library called QAPLIB [3]. Specifically, these instances, which range in size from $n = 12$ to $n = 40$, are Nug12, Nug16a, Nug20, Nug25, Nug30, Esc32a, and Lipa40a. All of these instances are expressed in the "Koopmans-Beckmann" form discussed in Section 1.1. We tested each instance

ten times using each of our five initial population methods (for a total of 350 trials). Then, for each instance, we compared the average quality for the initial population, each fifty generation, and the final solution (over the ten replications) across each of the different initial populations methods using an ANOVA F-test to check whether they are statistically significant different from each other. The results of these experiments are presented in Section 3.

## 2.1 GA Structure

As mentioned earlier, there are many choices to make when designing a GA. The structure of our GA, summarized in the flowchart of Figure 1, is heavily influenced by the work of [1]. Our GA is implemented in Python, where each individual solution is represent by a list containing a permutation of the numbers 1 to $n$ (where $n$ is the size of the QAP instance). For example, when $n = 4$, the solution represented by the list $[3,1,4,2]$ has facility 3 assigned to location site 1, facility 1 to site 2, facility 4 to site 3, and facility 2 to site 4. As a quick note about GA terminology, the permutation that represents a solution is often referred to as a "chromosome", the positions within the chromosome are called "genes", and the individual numbers in the chromosome are called "alleles". The objective function cost associate with a solution is referred to as that solution's "fitness", so a solution with a low cost has high fitness. In our algorithm, we maintain a "population" of 100 solutions. We discuss each individual element of our GA below. Please see the Supplemental Materials for the Python code for our GA.

*Initial Population.* We implemented five different methods for constructing the initial population of 100 solutions:

1. Random.
2. Random plus Local Search. (Random + LS)
3. Random plus Bad Local Search. (Random + Bad LS)
4. A Greedy Randomized Adaptive Search Procedure (GRASP).
5. A Greedy Randomized Adaptive Search Procedure plus Local Search. (GRASP + LS)

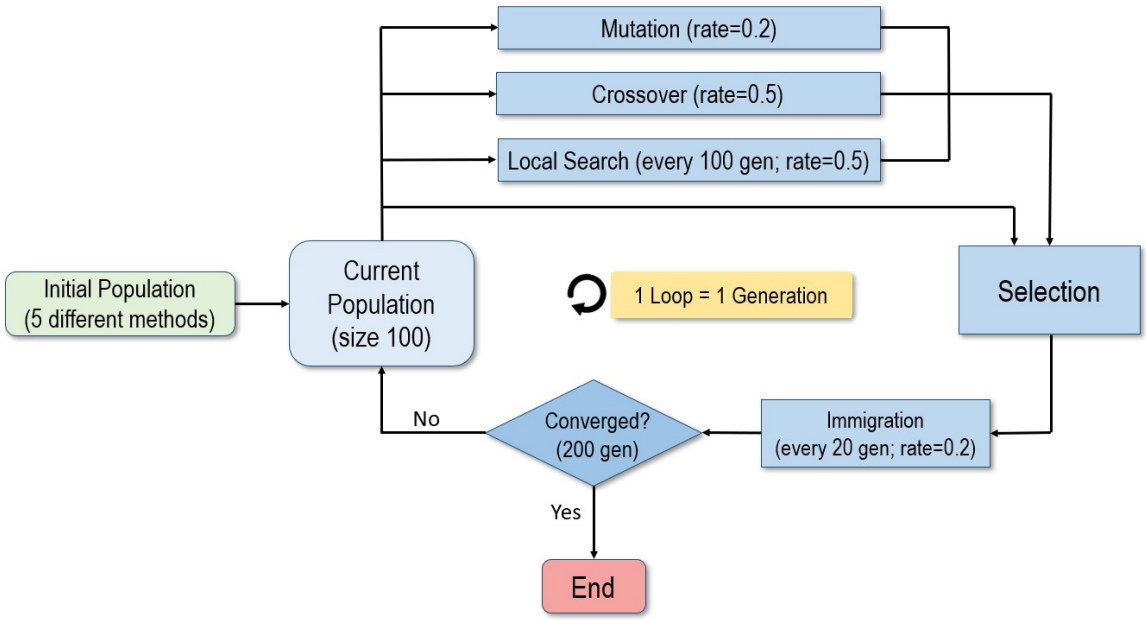


Figure 1: This flowchart illustrates the overall structure of our GA.

The *Random* method randomly selects 100 different solutions to make up the initial population. The *Random plus Local Search* method is similar, except that it replaces each of the 100 random solutions with its most fit " neighbor." Here, two solutions are defined to be neighbors if the only difference between them is that two alleles are swapped. For example, for a size $n = 6$ QAP, the solutions $[3,5,1,6,4,2]$ and $[3,6,1,5,4,2]$ are neighbors with the alleles 5 and 6 swapped. Note that every solution has $\frac{n(n-1)}{2}$ neighbors. The *Random plus Bad Local Search* method instead replaces each of the 100 random solutions with its *least* fit n eighbor. This method is used to create a bad initial population, which will help us determine whether or not the initial population actually matters.

The final two initial population methods use a procedure known as GRASP [12], which is one of the most well-known heuristics for finding good QAP solutions. GRASP uses a randomized greedy algorithm to sequentially assign facilities to location sites one at a time. The goal for each new assignment is to minimize the total additional cost of that new assignment with respect to all of the previously assigned facility/location pairs. As suggested in [1], our *GRASP* method only applies the

"construction" phase of the procedure outlined in [12], while our *GRASP plus Local Search* method mimics the full procedure of [12] by additionally applying the local search algorithm described in the previous paragraph.

In general, we would expect the ranking of our initial population methods (in terms of overall fitness, from worst to best) to be: Random plus Bad Local Search, Random, Random plus Local Search, GRASP, and finally GRASP plus Local Search. As we will see in Section 3, our experiments clearly showed that this was true. The main question, however, is whether these clear differences seen in the fitness of the initial populations will still be evident after the GA has completed its optimization.

*Mutation.* Every generation, 20% of the solutions undergo a simple swap mutation, where a solution mutates to a randomly chosen neighboring solution. For example, the solution $[3,4,1,2,6,5]$ might mutate to $[5,4,1,2,6,3]$. The new solution is added to the population. (We chose to also retain the original solution.)

*Crossover.* Every generation, 50% of the solutions are paired up to undergo what is known as an insert, or shift crossover. The crossover operation mimics reproduction in biological evolutionary systems, with two parent solutions combining to form offspring, or children. The child with the best fitness replaces the least fit of the two parents (i.e., that parent with the greatest cost). See [1] for a detailed description of the insert crossover scheme.

*Local Search.* Every 100 generations, 50% of solutions undergo a local search procedure, where the solution's most fit neighbor is added to the population. The original solution also remains in the population.

*Selection.* After mutation, crossover, and local search, there will be more than 100 solutions in the population (because mutation and local search generate new solutions without replacing the original ones). In the selection phase, all of the solutions in the population are ranked based on fitness, and only the top 100 solutions are selected to remain. The rest of the solutions are discarded. This is analogous to Darwin's ideas of natural selection and survival of the fittest.

*Immigration.* A common problem with GAs (and metaheuristics in general) is premature convergence to a local optimal solution. In order to mitigate this risk, we perform an immigration oper-

ation every 20 generations with the sole purpose of introducing diversity into the population. During immigration, the least fit 20% of the population is replaced by solutions that are created specifically to include facility/location assignments that have been underrepresented during the GA's execution up to that point. The idea is that these new solutions may introduce high-performing genetic material into the population that would have been difficult to encounter through mutation and crossover alone. See [1] for a detailed description of this procedure.

*Convergence Criterion.* There are many different convergence criteria that may be used to terminate a GA [10]. We chose to simply terminate our algorithm after 200 generations had passed, which for the instances we chose to test, seemed to reasonably balance computation time and the desire not to stop the algorithm while the GA is still making significant progress.

## 3 Computational Results

As mentioned in Section 2, we tested each of our five initial population methods on QAP test instances of size $n = 12$, $n = 16$, $n = 20$, $n = 25$, $n = 30$, $n = 32$, and $n = 40$. For each combination of test instance and initial population method, we ran ten replications of the GA. The graphs in Figure 2 summarize our findings. Each of the seven graphs represents one of the test instances, and the different colors represent the different initial population methods. The x-axis represents the generation (recall that our GA terminates after 200 generations) and the y-axis shows the average best cost. This average best cost is the cost (i.e., objective function value) of the *most fit* member of the population at the given generation, averaged over the ten replications. Since the QAP is a minimization problem, we would expect these plots all to be non-increasing functions, which is exactly what we observe in Figure 2.

There are several notable features about the graphs in Figure 2. First, looking at generation 0, we can observe that the initial population methods are behaving exactly as expected relative to each other, as discussed in Section 2.1. That is, the *GRASP plus Local Search* method produces the best initial populations, while the *Random plus Bad Local Search* method produces the worst. This relationship is most clearly seen in the $n = 32$ results. Meanwhile, the F-test also indicates that at

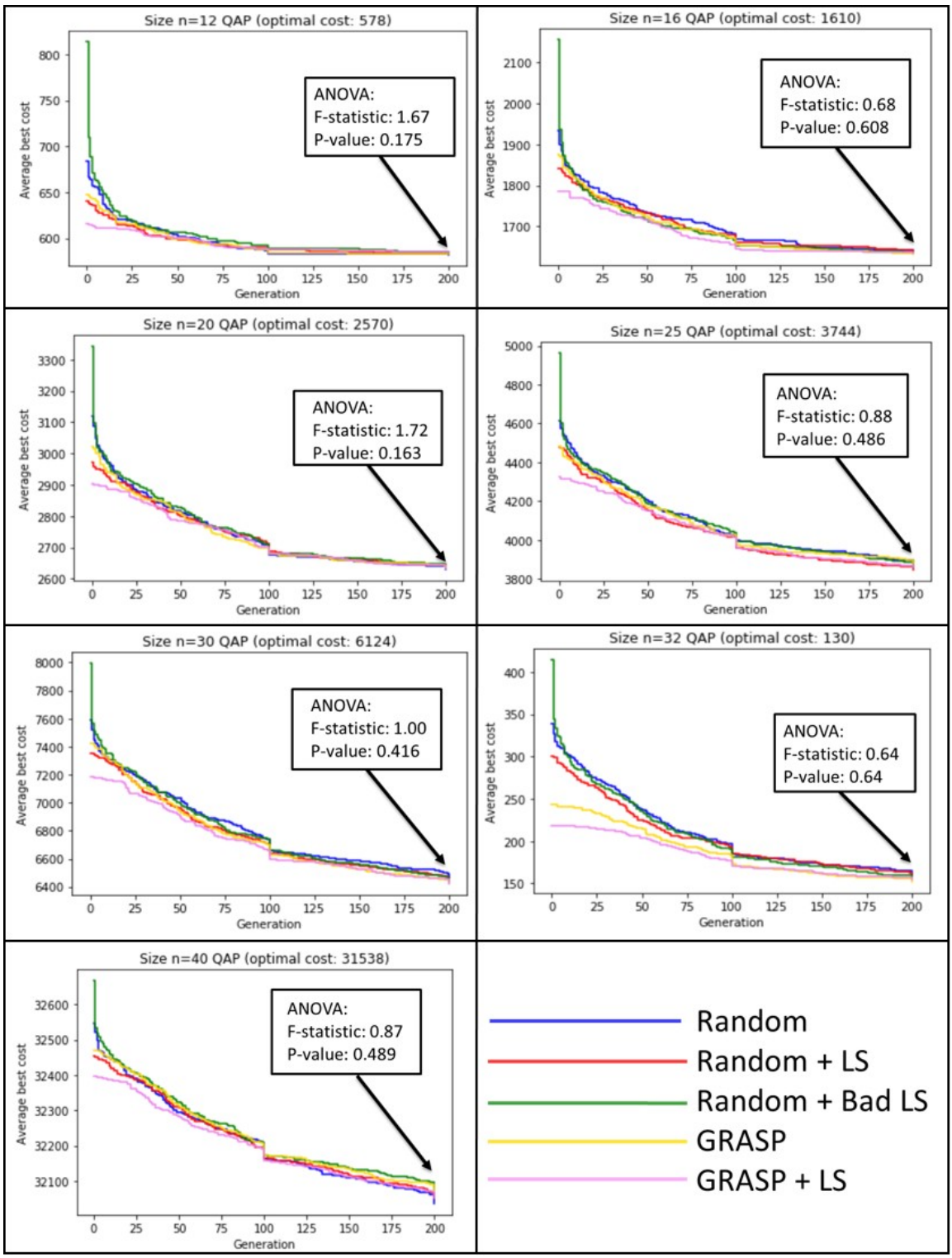


Figure 2: These graphs summarize the results of our experiments on QAP instances ranging from size $n = 12$ to $n = 40$.

least one group is statistically significant different from each other. Second, notice that every plot shows a sharp decrease at generation 100, which is due to the fact that our GA performs a special local search operation every 100 generations. Third, it is clear from these graphs that for every test instance, the different initial population methods produce initial populations with very different levels of fitness, but by generation 200, these differences are much less pronounced. The question we set out to answer, however, is whether or not these differences are still statistically significant at generation 200.

In order to test this, we performed an ANOVA F-test, with the null hypothesis being that there is no difference between the average best cost at generation 200 across the five different initial population methods. The F-statistics and corresponding p-values are shown in Figure 2. Note that the smallest p-value is 0.163, for the $n = 20$ test instance (we used a significance level of 0.05 for all tests). Therefore, we did not find sufficient evidence to conclude that there is a statistically significant difference in average best fitness across the five initial population methods after running our GA for 200 generations.

## 4 Conclusions

The QAP is one of the most well-known NP-hard combinatorial optimization problems. Its difficulty has led many practitioners to use metaheuristics, such as GAs, for solving large instances. For many types of optimization problems, the method used to form the initial population of a GA can have a large effect on the eventual results, but previous work suggested that this may not be true for the QAP. In the end, our experiments supported this previous report by finding no statistical evidence of a difference in the results of using a GA to solve the QAP with different initial population methods.

There are many avenues for future work related to this research. First, we could extend our study to include more QAP test instances (perhaps of larger size) and more initial population methods. We could also change the parameters and/or structure of the GA to see if this has any effect on the outcome of our experiments. Furthermore, we could extend this work by testing our hypothesis on

other NP-hard combinatorial optimization problems, such as the quadratic minimum spanning tree problem, or by testing features of other metaheuristics, such as simulated annealing and tabu search.

## 5 Acknowledgments

The author would like to thank the support of Bucknell University's Emerging Scholars Summer Research, Scholarship & Creativity Program for providing the funding for this project.

## References

[1] R.K. AHUJA, J.B. ORLIN, AND A. TIWARI, *A greedy genetic algorithm for the quadratic assignment problem*, Comput. Oper. Res., 27 (2000), pp. 917-934.

[2] R.E. BURKARD, E. ÇELA, P. PARDALOS, AND L. PITSOULIS, *The quadratic assignment problem*, Handbook of Combinatorial Optimization, 3 (1998), pp. 241-338.

[3] R.E. BURKARD, S.E. KARISCH, AND F. RENDL, *QAPLIB – A quadratic assignment problem library*, J. Global Optim., 10 (1997), pp. 391-403.

[4] R.E. BURKARD AND J. OFFERMANN, *Entwurf von schreibmaschinentastaturen mittels quadratischer zuordnungsprobleme*, Zeitschrift für Operations Research, 21 (1977), pp. B121-B132.

[5] E. ÇELA, *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer Academic Publishers, Dordrecht, 1998.

[6] P.A. DIAZ-GOMEZ AND D.F. HOUGEN, *Initial population for genetic algorithms: A metric approach*, in Proceedings of the 2007 International Conference on Genetic and Evolutionary Methods, Las Vegas, NV, 2007.

[7] J. DICKEY AND J. HOPKINS, *Campus building arrangement using TOPAZ*, Transportation Research, 6(1) (1972), pp. 59-68.

[8] A. ELSHAFEI, *Hospital layout as a quadratic assignment problem*, Operations Research Quarterly, 28(1) (1977), pp. 167-179.

[9] A. HAGHANI AND M.-C. CHEN, *Optimizing gate assignments at airport terminals*, Transportation Research Part A: Policy and Practice, 32(6) (1998), pp. 437-454.

[10] B.J. JAIN, H. POLHEIM, AND J. WEGENER, *On termination criteria of evolutionary algorithms*, in Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA, 2001.

[11] T. KOOPMANS AND M. BECKMANN, *Assignment problems and the location of economic activities*, Econometrica, 25(1) (1957), pp. 53-76.

[12] T. LI, P.M. PARDALOS, AND M.G.C. RESENDE, *A greedy randomized adaptive search procedure for the quadratic assignment problem*, in Quadratic Assignment and Related Problems, P.M. Pardalos and H. Wolkowicz, eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Providence, RI, 1994, pp. 237-261.

[13] E.M. LOIOLA, N.M. MAIA DE ABREU, P.O. BOAVENTURA-NETTO, P.M. HAHN, AND T. QUERIDO, *A survey for the quadratic assignment problem*, European J. Oper. Res., 176(2) (2007), pp. 657-690.

[14] A. MISEVICIUS, *A tabu search algorithm for the quadratic assignment problem*, Comput. Optim. Appl., 30 (2005), pp. 95-111.

[15] T. PENG, H. WANG, D. ZHANG, *Simulated annealing for the quadratic assignment problem: A further study*, Computers & Industrial Engineering, 31(3-4) (1996), pp. 925-928.